# Multiplication Algorithms

## Booth Algorithm for Multiplication

Booth algorithm multiplies two signed binary numbers in two’s complement notation

### Key Concept

The best way is to illustrate using examples

What about 011100?

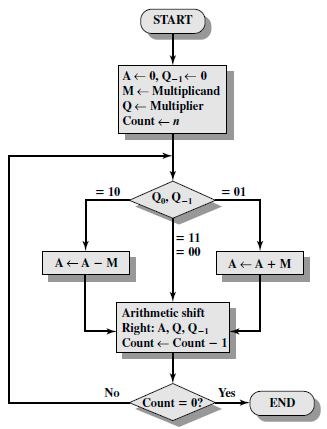
Similarly,

How can we build this in a systematic way?

* Add one extra bit as a least significant bit
* Initialize count = 0
* Scan the binary number from left to right, examine each two consecutive bits
  + ‘00’ or ‘11’ → increment count
  + ‘10’ → append -2count to the terms, increment count
  + ‘01’ → append 2count to the terms, increment count

### Multiply using the Previous Concept

Let

The following diagram does similar operations to multiply two signed binary numbers

### Why?

This algorithm is very fast and can work with signed numbers.

Given ,

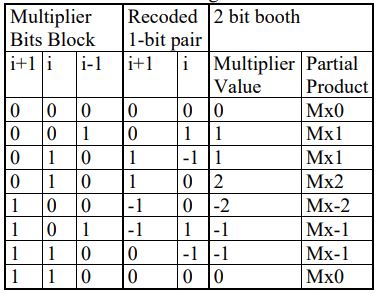
This algorithm performs N additions only.

### References

<https://courses.cs.washington.edu/courses/cse378/00sp/Sec5-1.htm>

## Modified Booth Algorithm

It is very similar to Booth Algorithm. However, the number of additions of partial product is reduced by half. Because it uses radix 4.



1. Extend the sign bit 1 position if necessary, to ensure that n is **even**.
2. Append a 0 to the right of the LSB of the multiplier.
3. According to the value of each vector, each Partial Product will be 0, +y, –y, +2y or –2y.

### Why?

It is faster than the original Booth algorithm.

### References

Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix-2) paper

# Addition Algorithms

## Carry Select adder

### Why?

Less delay time.

### References

https://en.wikipedia.org/wiki/Carry-select\_adder

## Ripple Carry adder

### Why?

Less hardware

### References

<http://www.circuitstoday.com/ripple-carry-adder>